

Toteutustekniikoita

Käsiteltäviä asioita:

1. Horn-klausuulien ratkaisumenetelmät
2. Dowling-Gallier-algoritmi
3. Stabiilien mallien hakutekniikkaa
4. Approksimaatioiden toteuttaminen
5. Hakualgoritmi stabiileille malleille

Neliöllinen ratkaisumenetelmä

1. Joukko Horn-klausuuleja S on toteutuva $\iff S' \not\models \perp$
(S' on joukkoa S vastaava ohjelmaklausuulien joukko).
2. $S' \not\models \perp \iff \perp \notin M_{S'}$.
3. Malli $M_{S'}$ voidaan laskea operaattorilla $T_{S'}$:
$$T_{S'} \uparrow 0, T_{S'} \uparrow 1, T_{S'} \uparrow 2, \dots$$
kunnes saavutetaan kiintopiste ($T_{S'} \uparrow n = T_{S'} \uparrow n - 1$).
4. Iteraatioita tarvitaan korkeintaan $|\text{HB}(S)| + 1$ kappaletta.
5. Joukon $T_{S'}(M)$ laskenta voidaan suorittaa mille tahansa $M \subseteq \text{HB}(S')$ lineaarisessa ajassa $\|S'\|$:n suhteen.
6. Aikakompleksisuus: $O(|\text{HB}(S')| \times \|S'\|)$.

1. Horn-klausuulien ratkaisumenetelmät

- Horn-klausuulit muodostavat keskeisimmän tehokkaasti toteutettavissa olevan osajoukon lauselogiikasta.
- Ratkaisuongelma: onko äärellinen joukko propositionaalisia Horn-klausuuleja S toteutuva?
- Kun tämä osataan ratkaista tehokkaasti, saamme myös tehokkaan tavan ratkaista ongelman

$$S \models Q_1 \vee \dots \vee Q_n$$

missä S on Horn-klausuulijoukko ja jokainen Q_i on literaalien konjunktio, jossa esiintyy korkeintaan yksi negatiivinen literaali.

Huomio. $S \models Q \iff S \cup \{\neg Q\}$ on toteutumaton.

2. Dowling-Gallier-algoritmi

- W.F. Dowling ja J.H. Gallier [1984]: *Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae*
- Idea: toteutetaan T_S -iteraatio tehokkaalla tietorakenteella siten, että kutakin atomia käsitellään vain kerran.
- Käytetään taulukoita, joita varten tarvitaan kullekin atomille yksikäsitteinen indeksi $\text{HB}(S) = \{P_0, P_1, \dots, P_{n-1}\}$.
- Jos indeksejä ei tiedetä etukäteen ja syötejoukko joudutaan jäsentämään, vaatii indeksien muodostaminen $O(\|S\| \times \log |\text{HB}(S)|)$ työn (symbolitaulu).
- Tämän jälkeen algoritmi toimii *linearisessa ajassa* $\|S\|$:n suhteen.

Tietorakenteet

Olkoon syötejoukossa n atomilauseetta ja m ohjelmaklausuulia.

Otetaan käyttöön seuraavat taulukot:

- $\text{val}[0 \dots n-1]$: atomien P_0, P_1, \dots, P_{n-1} totuusarvot (aluksi "true", jos P_i esiintyy ainoana (positiivisena) literaalina jossain ohjelmaklausuulissa, muutoin "false").
- $\text{occurs}[0 \dots n-1]$: niiden ohjelmaklausuulien indeksien lista, joissa atomi esiintyy negatiivisena.
- $\text{negcnt}[0 \dots m-1]$: klausuulin negatiivisten literaalien lukumäärä.
- $\text{poslit}[0 \dots m-1]$: klausuulin positiivinen literaali.

```

procedure minmodel;
while not empty(queue) and consistent do
  a := get(queue);
  clauses := occurs[a] ;
  while not empty(clauses) and consistent do
    clause := get(clauses); negcnt[clause] := negcnt[clause]-1;
    if negcnt[clause]=0 then
      n := poslit[clause];
      if val[n]=false then
        val[n] := true;
        if n=falseatom then consistent := false else push(n, queue);
      endif
    endif
  done
done

```

Algoritmi

Luetaan syötejoukko;

Alustetaan taulukot $\text{val}[0 \dots n-1]$, $\text{occurs}[0 \dots n-1]$,

$\text{negcnt}[0 \dots m-1]$ ja $\text{poslit}[0 \dots m-1]$;

$\text{consistent} := \text{true}$;

$\text{falseatom} :=$ atomin \perp indeksi;

$\text{queue} :=$ atomit, joille $\text{val}[i]$ aluksi true;

minmodel ;

/* Laskee syötteenä annetun ohjelmaklausuulijoukon pienimmän mallin taulukkoon val paitsi, jos \perp saa arvon tosi, jolloin keskeytetään. */

if consistent **then**

 print('satisfying model: '); printmodel

else print('unsat set of clauses')

endif

Esimerkki. Olkoon $S =$

$\{0 : P_0, 1 : P_1 \leftarrow P_0, 2 : P_2 \leftarrow P_1 \wedge P_0, 3 : P_3 \leftarrow P_1, 4 : \perp_5 \leftarrow P_2 \wedge P_4\}$.

Algoritmi laskee mallin $M_S = \{P_0, P_1, P_2, P_3\}$ seuraavasti:

$\text{val}[0] := \text{true}$ $\text{queue} := \langle 0 \rangle$ <hr style="width: 50%; margin: 5px 0;"/> $a := 0$ $\text{clauses} := [1, 2]$ $\text{clause} := 1$ $\text{negcnt}[1] := 0$ $\text{val}[1] := \text{true}$ $\text{queue} := \langle 1 \rangle$ $\text{clause} := 2$ $\text{negcnt}[2] := 1$	$a := 1$ $\text{clauses} := [2, 3]$ $\text{clause} := 2$ $\text{negcnt}[2] := 0$ $\text{val}[2] := \text{true}$ $\text{queue} := \langle 2 \rangle$ $\text{clause} := 3$ $\text{negcnt}[3] := 0$ $\text{val}[3] := \text{true}$ $\text{queue} := \langle 2, 3 \rangle$	$a := 2$ $\text{clauses} := [4]$ $\text{clause} := 4$ <hr style="width: 50%; margin: 5px 0;"/> $\text{negcnt}[4] := 1$ $a := 3$ $\text{clauses} := []$
---	---	---

Huomioita Dowling-Gallier-algoritmista

- Oikeellisuus: $\text{val}[i]=\text{true} \iff P_i \in \text{lfp}(T_S)$.
- Lineaarinen aikavaativuus suhteessa $\|S\|$:een.
 1. Kukin atomi päätyy jonoon (queue) korkeintaan kerran.
 2. Sisemmässä **while**-silmukassa tehdään työtä kokonaisuudessaan korkeintaan negatiivisten literaalien esiintymien lukumäärän verran.
- Algoritmin suorittaman hakutyyppi voidaan valita:
 - Jono \Rightarrow leveyshaku
 - Pino \Rightarrow syvyyshaku
- Samalla periaatteella voidaan laatia lineaarisen ajan toteutus yhden literaalin säännöstä (esim. Davis-Putnam -menetelmässä).

☞ Saadaan Davis-Putnam -menetelmää muistututtava algoritmi stabiilien mallien laskemista varten.

- Toteutustekniikkaa analysoidaan mm. seuraavassa artikkelissa: P. Simons, I. Niemelä, and T. Soinen [2002, AIJ]: *Extending and Implementing the Stable Model Semantics*
- Lisäksi on toteutettu muuttujia sisältävien ohjelmien instantiointi: T. Syrjänen [2004]: *Cardinality Constraint Programs*
T. Syrjänen [1998]: *Implementation of Local Grounding for Logic Programs with Stable Model Semantics*
- Ohjelmat `smodels` ja `lparse` ovat saatavilla verkossa osoitteella <http://www.tcs.hut.fi/Software/smodels/>

3. Stabiilien mallien hakutekniikkaa

- Tavoiteena laskea mahdollisimman tehokkaasti logiikkaohjelman P annetut kriteerit täyttävä(t) stabiili(t) malli(t).
- Täysiin joukkoihin perustuvan stabiilien mallien karakterisoinnin perusteella hakuavaruus muodostuu olennaisesti joukon $\text{NA}(P)$ (ohjelmassa P esiintyvät not-literaalit) osajoukoista.
- Rakennetaan asteittain stabiileja malleja rajaavaa osittaismallia (literaalijoukko FS) ja pyritään karsimaan hakuavaruutta.
 - (i) Malleja rajaavat oletukset tehdään yksi kerrallaan.
 - (ii) Jokaisessa hakupisteessä lasketaan approksimaatio kaikille stabiileille malleille, jotka toteuttavat senhetkiset oletukset ja tutkitaan, onko mahdollisesti saavutettu konflikti.

Approksimointiperiaatteet

- Laskettavat stabiilit mallit rajataan literaalijoukolla FS :
 - (i) Jos $A \in FS$, niin $A \in M$ laskettaville stabiileille malleille M .
 - (ii) Jos $\text{not } A \in FS$, niin $A \notin M$ laskettaville stabiileille malleille M (näitä vastaaville täysille joukoille F pätee $\text{not } A \in F$).
- Tämä ehdot voidaan ymmärtää ohjelman P stabiiliin mallin M ja literaalijoukon FS välisenä *yhteensopivuutena*.

Esimerkki. Tarkastellaan normaalia logiikkaohjelmaa

$$P = \{ A \leftarrow \text{not } B, B \leftarrow \text{not } A, C \leftarrow \text{not } D, \\ D \leftarrow \text{not } C, E \leftarrow \text{not } F, F \leftarrow \text{not } E \}.$$

Nyt esim. literaalijoukko $FS = \{A, \text{not } C\}$ on yhteensopiva ohjelman P stabiilien mallien $M_1 = \{A, D, E\}$ ja $M_2 = \{A, D, F\}$ kanssa.

- Approksimointi perustuu stabiilien mallien ala- ja ylärajaan:
 1. *Alaraja* $LB(P,FS) \supseteq FS$ on literaalijoukko, joka on yhteensopiva jokaisen literaalijoukon FS kanssa yhteensopivan ohjelman P stabiilin mallin $M \subseteq HB(P)$ kanssa.
 2. *Yläaraja* $UB(P,FS) \subseteq HB(P)$ on atomilauseiden joukko, joka sisältää jokaisen literaalijoukon FS kanssa yhteensopivan ohjelman P stabiilin mallin $M \subseteq HB(P)$.
- Approksimaatio **expand**(P,FS) on pienin joukko FS' , joka sisältää FS :n ja on suljettu seuraavien sääntöjen suhteen:
 - (i) Jos literaali $L \in LB(P,FS')$, niin $L \in FS'$.
 - (ii) Jos atomi $A \notin UB(P,FS')$, niin $\text{not } A \in FS'$.
- Approksimaatio **expand**(P,FS) saadaan laskemalla ala- ja ylärajoja toistuvasti ja täydentämällä literaalijoukkoa FS (kunnes saavutetaan kiintopiste periaatteiden (i) ja (ii) suhteen).

Alaraja

Määritelmä. Alaraja $LB(P,FS)$ on pienin joukko FS' , joka sisältää joukon FS ja on suljettu seuraavien periaatteiden suhteen:

- S1: Jos $H \leftarrow L_1 \wedge \dots \wedge L_n \in P_{FS'}$ ja $\{L_1, \dots, L_n\} \subseteq FS'$, $H \in FS'$.
- S2: Jos atomi H ei esiinny minkään ohjelman $P_{FS'}$ säännön seurauksena, niin $\text{not } H \in FS'$.
- S3: Jos $H \in FS'$ on täsmälleen yhden säännön $H \leftarrow L_1 \wedge \dots \wedge L_n \in P_{FS'}$ seurauksena, $\{L_1, \dots, L_n\} \subseteq FS'$.
- S4: Jos $\text{not } H \in FS'$, sääntö $H \leftarrow L_1 \wedge \dots \wedge L_n \in P_{FS'}$ ja $\{L_1, \dots, L_{i-1}, L_{i+1}, \dots, L_n\} \subseteq FS'$, niin $\bar{L}_i \in FS'$.
- S5: Jos jollekin atomille H sekä $H \in FS'$ että $\text{not } H \in FS'$, niin kaikki literaalit kuuluvat joukkoon FS' .

4. Approksimaatioiden toteuttaminen

- Tavoitteena on, että alaraja $LB(P,FS)$ ja yläaraja $UB(P,FS)$ voidaan laskea lineaarisessa ajassa.
- Vähintään yhtä tarkat rajat kuin WF-mallilla.
- Hakutilanne (osittaismalli FS) hyödynnettävä tehokkaasti.
- Oletuksilla FS vielä (mahdollisesti) käytettävissä olevat säännöt:

$$P_{FS} = \{ H \leftarrow B_1 \wedge \dots \wedge B_m \wedge \text{not } B_{m+1} \wedge \dots \wedge \text{not } B_n \in P \mid \\ FS \cap \{ \text{not } B_1, \dots, \text{not } B_m, B_{m+1}, \dots, B_n \} = \emptyset \}.$$

- Määritellään literaalin L komplementti \bar{L} normaaliin tapaan:

$$\bar{\bar{A}} = \text{not } A \text{ ja } \overline{\text{not } A} = A.$$
- Nyt $P_{FS} = \{ H \leftarrow L_1 \wedge \dots \wedge L_n \in P \mid FS \cap \{ \bar{L}_1, \dots, \bar{L}_n \} = \emptyset \}.$

Esimerkki. Tarkastellaan ohjelmaa

$$P_1 = \{ A \leftarrow \text{not } B \text{ (R1)}, C \leftarrow A \text{ (R2)}, \\ B \leftarrow A \wedge \text{not } C \wedge \text{not } D \text{ (R3)}, D \leftarrow C \wedge \text{not } E \text{ (R4)} \}.$$

- Alarajan $LB(P_1, \{ \text{not } B \})$ laskenta:

Literaalit	Periaate	Käytettävissä olevat säännöt
$\text{not } B$	oletus	R1, R2, R3, R4
A	S1	R1, R2, R3, R4
C	S1	R1, R2, R4
$\text{not } E$	S2	R1, R2, R4
D	S1	R1, R2, R4

- Approksimaatio **expand**($P_1, \{ \text{not } B \}) = \{ \text{not } B, A, C, \text{not } E, D \}$ määrää jo stabiilin mallin $\{A, C, D\}$ (johon atomi B ei kuulu).




Esimerkki. Tarkastellaan edelleen ohjelmaa

$$P_1 = \{ A \leftarrow \text{not } B \text{ (R1)}, C \leftarrow A \text{ (R2)}, \\ B \leftarrow A \wedge \text{not } C \wedge \text{not } D \text{ (R3)}, D \leftarrow C \wedge \text{not } E \text{ (R4)} \}.$$

► Alarajan $\text{LB}(P_1, \{B\})$ laskenta:

Literaalit	Periaate	Käytettävissä olevat säännöt
B	oletus	R2, R3, R4
$A, \text{not } C, \text{not } D$	S3	R2, R3
C	S1	R2
kaikki literaalit	S5	

► Täten myös $\text{expand}(P_1, \{B\})$ antaa kaikki literaalit
 ohjelmalla P_1 ei ole stabiilia mallia M siten, että $B \in M$.

Esimerkki. Lasketaan samaiselle ohjelmalle

$$P_2 = \{ A \leftarrow \text{not } B, B \leftarrow \text{not } A, C \leftarrow \text{not } A, \\ D \leftarrow \text{not } C, E \leftarrow \text{not } D \}$$

approksimaatiot $\text{expand}(P_2, \{A\})$ ja $\text{expand}(P_2, \{\text{not } A\})$.

- $\text{LB}(P_2, \{A\}) = \{A, \text{not } B, \text{not } C, D, \text{not } E\} = \text{expand}(P_2, \{A\})$.
- $\text{LB}(P_2, \{\text{not } A\}) = \{\text{not } A, B, C, \text{not } D, E\} = \text{expand}(P_2, \{\text{not } A\})$.

Esimerkki. Ohjelmalle $P_3 = \{A \leftarrow \text{not } B, B \leftarrow B\}$ alaraja $\text{LB}(P_3, \emptyset) = \emptyset$, mutta yläraja $\text{UB}(P_3, \emptyset) = \{A\}$.

Näin voidaan päätellä $\text{not } B$ (kts. expand). Tästä seuraa, että $\text{expand}(P_3, \emptyset) = \{A, \text{not } B\}$, koska $\text{LB}(P_3, \{\text{not } B\}) = \{A, \text{not } B\}$.



Yläraja

Määritelmä. Yläraja $\text{UB}(P, FS)$ on pienin mallin ohjelmalle P' , joka saadaan poistamalla not -litteraalit ohjelmasta PF_S .

Esimerkki. Tarkastellaan ohjelmaa

$$P_2 = \{ A \leftarrow \text{not } B, B \leftarrow \text{not } A, C \leftarrow \text{not } A, \\ D \leftarrow \text{not } C, E \leftarrow \text{not } D \}$$

Lasketaan ohjelmalle P_2 esim. seuraavat ylärajat:

- $\text{UB}(P_2, \emptyset) = \{A, B, C, D, E\}$
(koska lisäksi myös $\text{LB}(P_2, \emptyset) = \emptyset$, saadaan $\text{expand}(P_2, \emptyset) = \emptyset$).
- $\text{UB}(P_2, \{A\}) = \{A, D, E\}$.
- $\text{UB}(P_2, \{\text{not } A\}) = \{A, B, C, D, E\}$.

Ala- ja ylärajojen toteuttaminen

- Ala- ja ylärajoille voidaan antaa lineaarisen ajan toteutukset (käyttämällä Dowling-Gallier -tyyppisiä tietorakenteet).
- Tarkkuus on vähintään sama kuin WF-mallilla: $\text{expand}(P, \emptyset)$ antaa ohjelman P WF-mallin.
- Hakutilanne (osittaismalli FS) hyödynnetään dynaamisesti.

Esimerkki. Ohjelman $P = \{A \leftarrow \text{not } B, B \leftarrow \text{not } A, A \leftarrow B\}$ tapauksessa approksimaatioksi $\text{expand}(P, \{B\})$ saadaan $\{A, B, \text{not } A, \text{not } B\}$.

Ohjelman $P \cup \{B\}$ WF-malli on $\{A, B\}$ (konflikti ei tule esille!)

5. Hakualgoritmi stabiileille malleille

```

function smodels( $P, FS, \varphi$ ): boolean;
 $FS' := \text{expand}(P, FS)$ ;
if conflict( $P, FS'$ ) returns true then return false
else if NA( $P$ ) is covered by  $FS'$  then return test( $P, FS', \varphi$ )
else
   $\chi := \text{choose}(P, FS')$ ;
  if smodels( $P, FS' \cup \{\chi\}, \varphi$ ) returns true then
    return true
  else return smodels( $P, FS' \cup \{\bar{\chi}\}, \varphi$ )
  end if
end if

```

Esimerkki. Tutkitaan, kuuluuko atomi E ohjelman P kaikkiin stabiileihin malleihin etsimällä vastamalli, johon E ei kuulu.

Aloitetaan osittaismallista $FS = \{\text{not } E\}$
(valintafunktio **test** palauttaa aina arvon true).

$$P = \{ \begin{array}{ll} A \leftarrow \text{not } B, & \text{not } E \\ B \leftarrow \text{not } A, & D \\ C \leftarrow A \wedge \text{not } B, & \text{not } C \\ C \leftarrow B \wedge \text{not } A, & A \mid \text{not } A \\ D \leftarrow \text{not } C, & \text{not } B \mid B \\ E \leftarrow \text{not } D \} & \begin{array}{l} C \\ \times \end{array} \mid \begin{array}{l} C \\ \times \end{array}$$

 Kyllä kuuluu.

Funktioiden tehtävät:


- Funktiota **test** käytetään löytyneiden mallien valintaan (parametri φ antaa valintakriteerit).
- Funktio **conflict**(P, FS) tarkistaa, sisältääkö saatu approksimaatio FS konfliktin ($\{A, \text{not } A\} \subseteq FS$ jollekin $A \in \text{HB}(P)$).
- Funktio **choose** toteuttaa hakuheuristiikan (paluuarvona literaali).
- Funktio **smodels** suorittaa *fokusoitua mallinetsintää*:
 $\text{smodels}(P, FS, \varphi)$ palauttaa arvon true
 $\iff \exists P$:n stabiili malli $M = \text{Dcl}(P, \{\text{not } A \mid \text{not } A \in FS'\})$,
 joka on yhteensopiva FS :n kanssa ($FS \subseteq FS'$) ja
 jolle **test**(P, FS', φ) palauttaa arvon true.

Esimerkki. Etsitään ohjelman

$$P = \{ \begin{array}{l} A \leftarrow C \wedge \text{not } B, \\ B \leftarrow \text{not } A, \\ C \leftarrow \text{not } D, \\ D \leftarrow \text{not } A \} \end{array}$$

stabiilit mallit (alussa $FS = \emptyset$ ja **test** palauttaa aina true):

$$\begin{array}{l|l} A & \text{not } A \\ \text{not } D & B \\ C & D \\ \text{not } B & \text{not } C \end{array}$$

 Stabiilit mallit $M_1 = \text{Dcl}(P, \{\text{not } B, \text{not } D\}) = \{A, C\}$ ja
 $M_2 = \text{Dcl}(P, \{\text{not } A, \text{not } C\}) = \{B, D\}$.

Heuristiikka

- Lookahead-tekniikka: jokaiselle atomille $A \in \text{NA}(P)$, johon osittaismalli FS ei ota kantaa ($A \notin FS$ ja $\text{not } A \notin FS$), lasketaan $FS_1 = \text{expand}(P, FS \cup \{A\})$ ja $FS_2 = \text{expand}(P, FS \cup \{\text{not } A\})$
- Saadaan tarkempi alaraja:
Jos FS_1 sisältää konfliktin, lisätään $\text{not } A$ joukkoon FS ja jos FS_2 sisältää konfliktin, lisätään A joukkoon FS .
- FS_1 and FS_2 antavat arvion jäljellä olevasta hakuavaruudesta.
- Heuristiikka: valitaan atomi, jolla on pienin arvioitu jäljellä oleva hakuavaruus.

Oppimistavoitteet

- Tunnet Dowling-Gallier-algoritmin toimintaperiaatteen ja osaat tarvittaessa simuloida algoritmin toimintaa annetulle ohjelmalle.
- Osaat sovittaa Davis-Putnam-tyyppisen hakualgoritmin stabiilien mallien tapaukseen.
- Osaat hakea annetulle logiikkaohjelmalle stabiilit mallit.
- Tunnet joitain approksimointiperiaatteita ja pystyt hyödyntämään niitä vaihtoehtojen karsimiseen laskiessasi stabiileja malleja.

Arviointia

- Logiikkaohjelmat ja stabiilit mallit ilmaisuvoimainen rajoiteohjelmointiparadigma.
(Esim. Boolean rajoitteet/lauselogiikka erikoistapaus).
- Hakuavaruuden rakenne poikkeaa lauselogiikan toteutuvuusongelmasta:
 - (i) Vain not -litteraalit lisäävät hakuavaruutta.
 - (ii) Säännöt suunnattuina: kerrostuneisuus rajaa hakuavaruutta.
 ➤ Enemmän vapautta tietämyksen esittämiseen
- Toteutustekniikat parantuneet ratkaisevasti: mm. edellä kuvatun algoritmin tilavaativuus on lineaarinen ohjelman kokoon nähden.